УДК 37.026.4

ПОСТРОЕНИЕ ГЕОМЕТРИЧЕСКИХ ФРАКТАЛОВ В СКМ MAPLE

А.А. Агафонов[1]

1 *a.a.agathonov@gmail.com*; Казанский (Приволжский) федеральный университет

Приведены примеры построения некоторых известных геометрических фракталов в СКМ Maple

Ключевые слова: геометрические фракталы, программирование, СКМ Maple.

История фракталов началась с геометрических фракталов, которые исследовались математиками в 19 веке. Геометрические фракталы являются самыми наглядными, так как видна их самоподобная структура. В двухмерном случае их получают с помощью некоторой ломаной, называемой *генератором*. За один шаг алгоритма каждый из отрезков (*инициаторы*), составляющих ломаную, заменяется на ломаную-генератор, в соответствующем масштабе. В результате бесконечного повторения этой процедуры (*правило рекурсии*), получается геометрический фрактал. Примерами таких фракталов являются треугольник Серпинского, снежинка Коха, кривая Леви и многие другие. В графике геометрические фракталы применяются для получения изображений деревьев, кустов, береговых линий и т.д.

Кривая Леви

Чтобы получить Кривую Леви, на отрезке AB (инициатор) построим равнобедренный прямоугольный треугольник ACB так, чтобы угол $\angle ACB$ был прямым, после чего удалим гипотенузу (Рис. 1). Полученная ломаная линия из двух отрезков AC и CB представляет генератор, по которому строится Кривая Леви. Применяя генератор к каждому новому отрезку, получим вторую итерацию кривой, состоящее из 4 отрезков. Дальнейшее применяем генератор ко всем отрезкам кривой даст в пределе Кривую Леви.

Вычислим координаты точки C по координатам известных точек $A(x_A, y_A)$ и $B(x_B, y_B)$:

$$\begin{aligned}
&\overline{AC} = \overline{AO} + \overline{OC};\\
&\overline{AB} = (x_B - x_A, y_B - y_A); \quad \overline{AO} = \frac{1}{2}\overline{AB} = \frac{1}{2}(x_B - x_A, y_B - y_A);\\
&|\overline{OC}| = |\overline{AO}|, \quad \overline{OC} \perp \overline{AO} \quad \Rightarrow \quad \overline{OC} = \frac{1}{2}(-(y_B - y_A), x_B - x_A);
\end{aligned} \tag{1}$$

$$C = \left(x_A + \frac{1}{2}(x_B - x_A) + \frac{1}{2}(-(y_B - y_A)),\ y_A + \frac{1}{2}(y_B - y_A) + \frac{1}{2}(x_B - x_A) \right) \tag{2}$$

В СКМ Maple удобно работать с координатами точек и компонентами векторов как со списками чисел. Тогда список координат точки C легко найти, используя формулу (2): `A=[x_A,y_A], B=[x_B,y_B], C=A+1/2*(B-A)+1/2*[-(y_B-y_A),x_B-x_A]`.

Построение фрактала будем производить рекурсивно: возьмем список, содержащий вложенные списки координат начальных точек кривой; переберем все элементы списка и между каждой парой точек вставим координаты точек, вычисленных по формуле (2); переберем полученный список и между каждой парой точек

вновь вставим новые точки; проделаем процедуру n раз:

$$[\ldots, X_{i-1}, X_i, \ldots] \quad \rightarrow \quad [\ldots, X_{i-1}, Y, X_i, \ldots].$$

Далее строим кривую, соединяя все точки списка попарно отрезками прямых.

```
1  _Levy:=proc(lst,n)
2  local shifted,temp,result,i:
3    if n=0 then
4      lst:
5    else
6      temp:=_Levy(lst,n-1):
7      result:=[temp[1]]:
8      for i from 2 to nops(temp) do
9        shifted:=temp[i-1]+1/2*(temp[i]-temp
             [i-1])+1/2*[-(temp[i]-temp[i-1])
             [2],(temp[i]-temp[i-1])[1]]:
10       result:=[op(result),shifted,temp[i
             ]]:
11     end do:
12     result:
13   end if:
14 end proc:
15
16 > _Points:=[[0,0],[1,0]]:
17 result:=_Levy(Points,11):
18 display([seq(line(result[i],result[i+1],
       color=black),i=1..nops(result)-1)],
       scaling=constrained,axes=none);
```

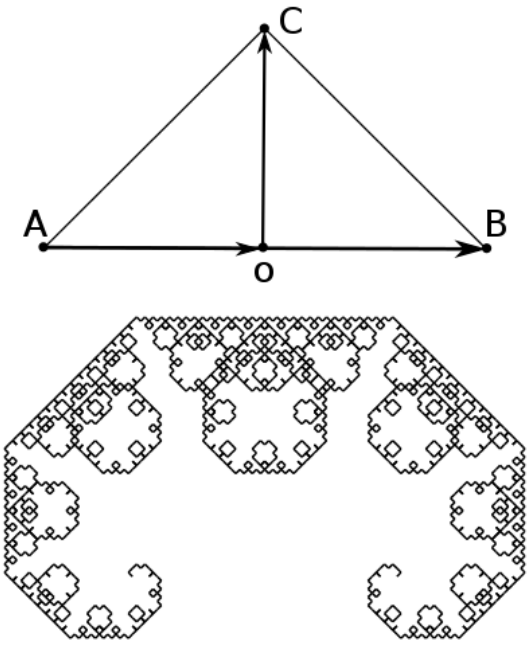


Рис. 1. Кривая Леви, 11 итераций.

Кривая Дракон Хартера-Хейтуэя

Если немного изменить генератор кривой Леви, то можно получить фрактал “Дракон Хартера-Хейтуэя”. Для этого необходимо вектор $\overline{OC}$ каждой следующей пары точек поворачивать на угол π. Этого легко добиться, умножением компонент вектора $\overline{OC}$ в формуле (2) на коэффициент $(-1)^i$, где i – порядковый номер сегмента кривой. Заменим 9-ую строку кода кривой Леви и получим кривую “Дракон Хартера-Хейтуэя” (Рис. 2).

```
9  shifted:=temp[i-1]+1/2*(temp[i]-temp[i-1])
       +(-1)^i*1/2*[-(temp[i]-temp[i-1])[2],(
       temp[i]-temp[i-1])[1]]:
```

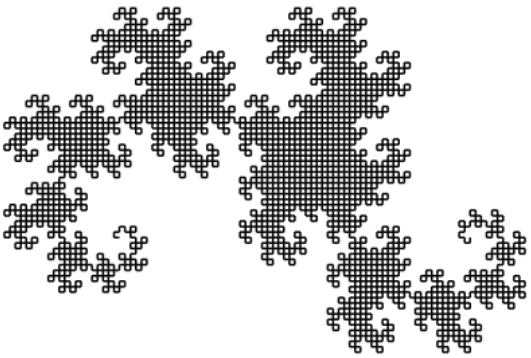


Рис. 2. Кривая Дракон, 12 итераций.

Снежинка Коха

Построение кривой начинается с отрезка AB (инициатор), $n = 0$. Далее отрезок заменяется на генератор $ACDEB$ (Рис. 3) – это кривая из четырех прямолинейных звеньев, каждое длиной по $\frac{1}{3}AB$, $n = 1$. Для получения каждой последующей итерации, все отрезки новой кривой заменяются генераторами. При n стремящемся к

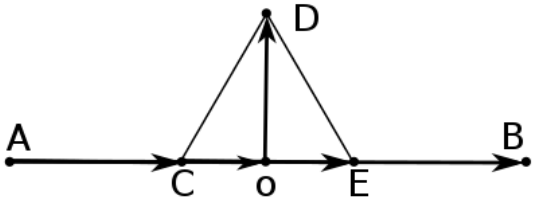
бесконечности кривая Коха становится фрактальным объектом.

Вычислим координаты всех необходимых точек генератора снежинки Коха:

$$
\begin{aligned}
&\overline{AB} = (x_B - x_A, y_B - y_A); \quad \overline{AO} = \frac{1}{2}\overline{AB} = \frac{1}{2}(x_B - x_A, y_B - y_A);\\
&\overline{AC} = \frac{1}{3}\overline{AB} = \frac{1}{3}(x_B - x_A, y_B - y_A); \quad \overline{AE} = \frac{2}{3}\overline{AB} = \frac{2}{3}(x_B - x_A, y_B - y_A);\\
&|\overline{OD}| = \frac{\sqrt{3}}{6}|\overline{AB}|, \quad \overline{OD} \perp \overline{AB} \quad \Rightarrow \quad \overline{OD} = \frac{\sqrt{3}}{6}(-(y_B - y_A), x_B - x_A);\\
&C = \left(x_A + \frac{1}{3}(x_B - x_A),\ y_A + \frac{1}{3}(y_B - y_A)\right);\\
&D = \left(x_A + \frac{1}{2}(x_B - x_A) + \frac{\sqrt{3}}{6}(-(y_B - y_A)),\ y_A + \frac{1}{2}(y_B - y_A) + \frac{\sqrt{3}}{6}(x_B - x_A)\right);\\
&E = \left(x_A + \frac{2}{3}(x_B - x_A),\ y_A + \frac{2}{3}(y_B - y_A)\right).
\end{aligned}
\tag{3}
$$

Для построения кривой Коха нужно взять произвольный начальный отрезок или ломаную и рекурсивно для каждой пары точек списка вычислять и вставлять координаты промежуточных трех точек (формула (3)):

$$[\ldots, X_{i-1}, X_i, \ldots] \quad \rightarrow \quad [\ldots, X_{i-1}, Y_1, Y_2, Y_3, X_i, \ldots]$$



Чтобы построить кривую Коха, нужно изменить 9 и 10 строки программного кода процедуры построения кривой Леви на следующие:

```
9  point1:=temp[i-1]+1/3*(temp[i]-temp[i-1]):
10 point2:=temp[i-1]+1/2*(temp[i]-temp[i-1])+
       sqrt(3.)/6*[-(temp[i]-temp[i-1])[2],(
       temp[i]-temp[i-1])[1]]:
11 point3:=temp[i-1]+2/3*(temp[i]-temp[i-1]):
12 result:=[op(result),point1,point2,point3,
       temp[i]]:
```

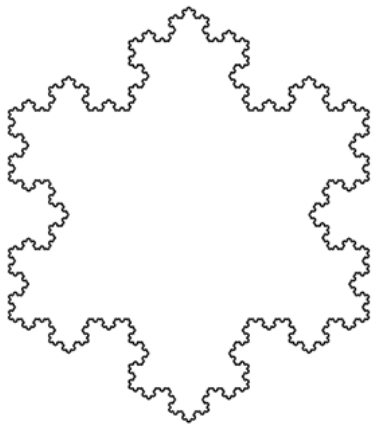


Рис. 3. Снежинка Коха, 5 итераций.

Снежинка

Для построения фрактальной снежинки необходимо построить k-лучевую звезду с центром в заданной точке (список `cntr` в программном коде) и длинной лучей равной r. Далее, конец каждого луча используется в качестве центра звезд следующего поколения, длина лучей которых меньше родительской. Получаем k штук звезд второго поколения. Далее на концах всех лучей всех звезд первого поколения строим звезды второго поколения и т.д. Ниже приведен программный код генерации фрактальной снежинки и пример трех итераций фрактала.

```
_snowflake:=proc(cntr,r,k,n)
global graph:
local t,i,temp:
  t:=2.*Pi/k;
  for i from 1 to k do
    temp:=[cntr[1]+r*cos(i*t),cntr[2]+r*sin(i*t
        )]:
    graph:=[op(graph),line(cntr,temp)]:
    if n>=1 then _snowflake(temp,.25*r,k,n-1)
        end if:
  end do:
end proc:

> cntr:=[0,0]:
graph:=[]:
_snowflake(cntr,10.,8,3):
display(graph,scaling=constrained,axes=none);
```

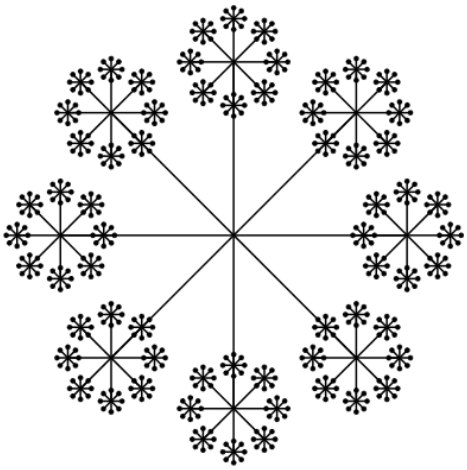


Рис. 4. Фрактальная снежинка, 3 итерации.

Дерево Пифагора

Для построения Дерева Пифагора используется генератор в виде прямоугольного треугольника с квадратом на гипотенузе. Применяя этот генератор к самому себе, получим первую итерацию с двумя новыми треугольниками. Продолжая процесс, получим новые поколения, для которых количество треугольников увеличивается каждый в два раза. В пределе этого процесса «вырастет» Дерево Пифагора.

Сначала рассмотрим процедуру построения одного квадрата по координатам вершины A (список `crnr` в программном коде), длине стороны AB (`side`) и углу поворота α (`angle`) стороны AB вокруг вершины A. Ниже приведены процедура и поясняющий рисунок.

```
DrawRect:=proc(crnr,side,angle)
local crnr2,crnr3,crnr4,line1,line2,
    line3,line4:
  crnr2:=crnr+side*[cos(angle),sin(
      angle)]:
  crnr3:=crnr+sqrt(2.)*side*[cos(angle
      -Pi/4),sin(angle-Pi/4)]:
  crnr4:=crnr+side*[cos(angle-Pi/2),
      sin(angle-Pi/2)]:
  line1:=line(crnr,crnr2):
  line2:=line(crnr2,crnr3):
  line3:=line(crnr3,crnr4):
  line4:=line(crnr4,crnr):
  line1,line2,line3,line4:
end proc:
```

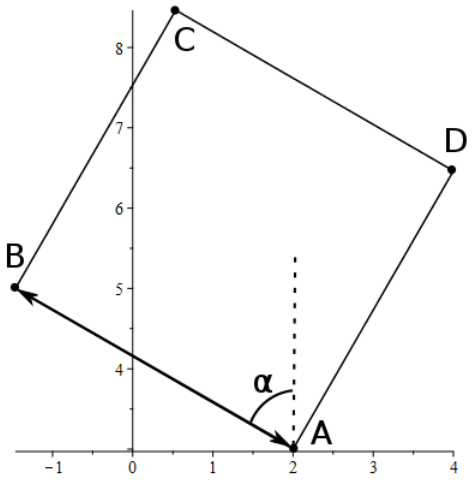


Рис. 5. К построению дерева Пифагора.

Теперь задача построения дерева Пифагора сводится к рекурсивному построению потомков по известным параметрам родительского квадрата. Приведем программный код построения симметричного дерева Пифагора (Рис. 6).

```
_PythagorasTree:=proc(crnr,side,angle,n)
```

```
2 global graph:
3  graph:=[op(graph),DrawRect(crnr,side,angle)]:
4  if n>=1 then
5   _PythagorasTree(crnr+side*[cos(angle),sin(angle)],side/sqrt(2.),angle+Pi/4.,n-1):
6   _PythagorasTree(crnr+side*[cos(angle),sin(angle)]+side/sqrt(2.)*[cos(angle-Pi/4.)
        ,sin(angle-Pi/4.)],side/sqrt(2.),angle-Pi/4.,n-1):
7  end if:
8 end proc:
```

При использовании неравнобедренного треугольника в качестве генератора, получим несимметричное дерево Пифагора (Рис. 7).

```
5 _PythagorasTree(crnr+side*[cos(angle),sin(angle)],side*sqrt(3.)/2.,angle+Pi/6.,n-1)
      :
6 _PythagorasTree(crnr+side*[cos(angle),sin(angle)]+side*sqrt(3.)/2.*[cos(angle-Pi
     /3.),sin(angle-Pi/3.)],side/2.,angle-Pi/3.,n-1):
```

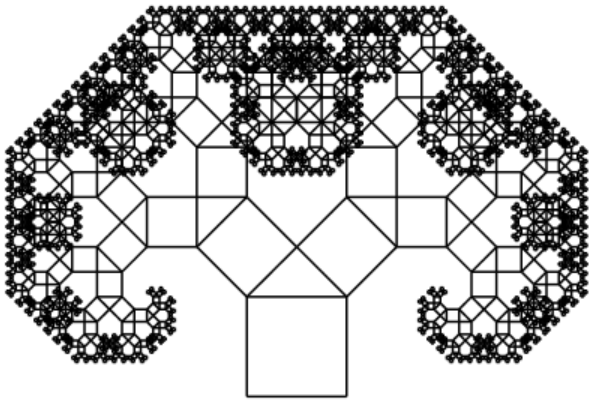


Рис. 6. Дерево Пифагора, 10 итераций.

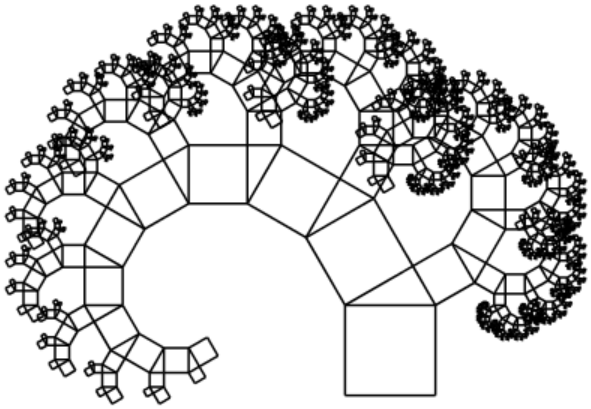


Рис. 7. Дерево Пифагора, 10 итераций.

Фрактальное дерево

Рассмотрим генерацию фрактального дерева. Для построения одного сегмента дерева необходимо построить отрезок по координатам начала отрезка (список `root` в программном коде), длине отрезка (`side`) и углу поворота (`angle`) отрезка. Далее строим два потомка меньшей длинны из конца родительского отрезка. В примере ниже один потомок отклоняется от направления родительского сегмента на угол $\pi/4$ против хода часовой стрелки, другой – на угол $\pi/6$ по ходу. Рекурсивно повторяем процедуру (Рис. 8).

Во втором примере потомки отклоняются на случайные углы в диапазоне $[0, \pi/4]$ от направления родительского сегмента, и длины потомков выбираются случайно в диапазоне $[0.65, 0.95]$ длины родительного сегмента (Рис. 9).

```
_tree:=proc(root,side,angle,n)
global graph:
local temp:
  temp:=root+side*[cos(angle),sin(
      angle)]:
  graph:=[op(graph),line(root,temp)]:
  if n>=1 then
    _tree(temp,.7*side,angle+Pi/4.,n
        -1):
    _tree(temp,.7*side,angle-Pi/6.,n
        -1):
  end if:
end proc:
```

```
randomize:
Deflection:=rand(0...1000)/1000*Pi/4.:
Scale:=rand(0...1000)/1000*.3+.65:
_tree:=proc(root,side,angle,n)
global graph:
local temp:
  temp:=root+side*[cos(angle),sin(
      angle)]:
  graph:=[op(graph),line(root,temp)]:
  if n>=1 then
    _tree(temp,Scale()*side,angle+
        Deflection(),n-1):
    _tree(temp,Scale()*side,angle-
        Deflection(),n-1):
  end if:
end proc:
```

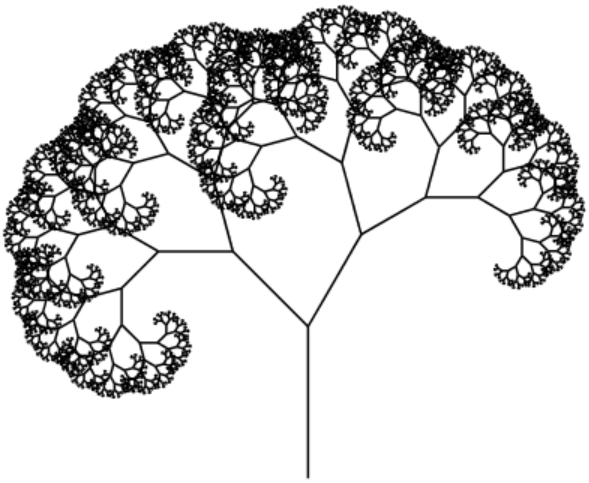


Рис. 8. Фрактальное дерево с фиксированными углами отклонения потомков.

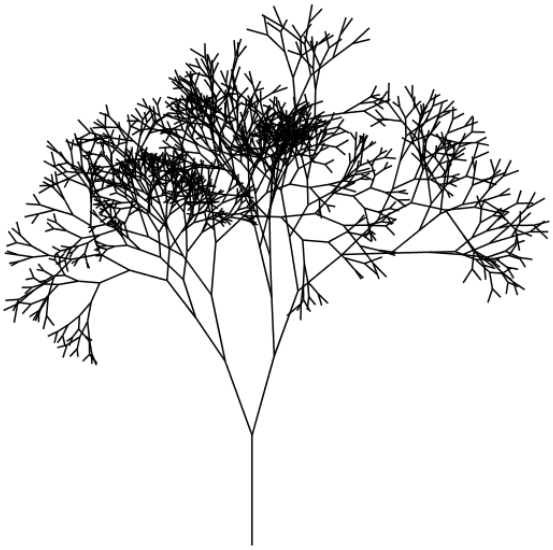


Рис. 9. Фрактальное дерево со случайными углами отклонения потомков.

Рассмотрим процедуру построения фрактального дерева, в котором каждое следующее поколение сегментов отличается от родителей по толщине и расцветке. Углы поворота и длины потомков также выбираются случайным образом (Рис. 10).

```
randomize:
Deflection:=rand(0...1000)/1000*Pi/3.:
Scale:=rand(0...1000)/1000*.3+.6:
_tree:=proc(root,side,angle,n)
global graph:
local temp:
 if side > 1.8 then
  temp:=root+side*[cos(angle),sin(angle)]:
  graph:=[op(graph),line(root,temp,color=COLOR(RGB,0,1-7/n,0),thickness=trunc(1+6/
      sqrt(n)))]:
  _tree(temp,Scale()*side,angle+(Pi/10.+Deflection()*n/15),n+1):
  _tree(temp,Scale()*side,angle-(Pi/10.+Deflection()*n/15),n+1):
```

```
 end if:
end proc:

> cntr:=[0,0]:
graph:=[]:
_tree(cntr,80.,Pi/2.,1):
display(graph,scaling=constrained,axes=none);
```

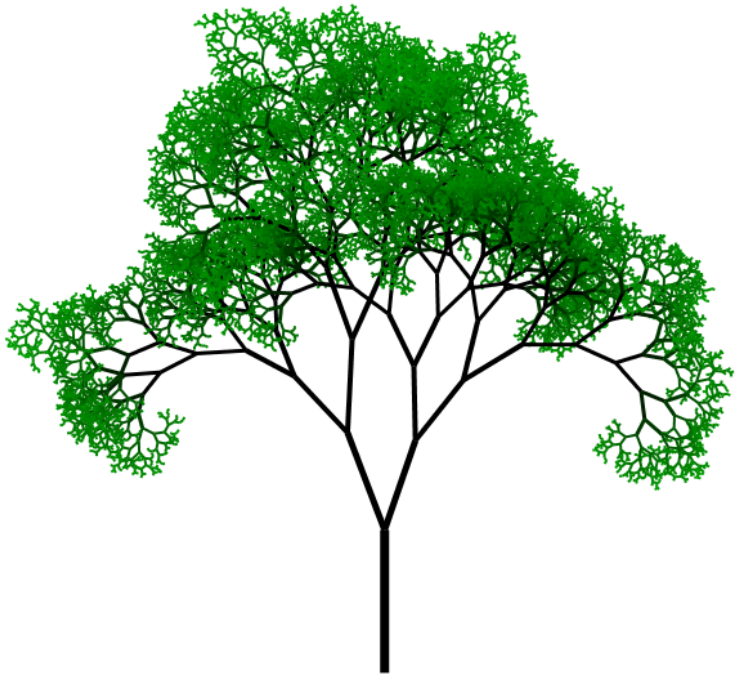


Рис. 10. Фрактальное дерево.

В заключении приведем без объяснения программный код построения фрактальной ветки (Рис. 11) и фрактального папоротника (Рис. 12).

Фрактальная ветка

```
_branch:=proc(root,side,angle,n)
global graph:
local end1,end2,end3,root2,root3:
  end1 :=root +side*[cos(angle),sin(angle)]:
  root2:=root +1/6*side*[cos(angle),sin(angle)]:
  end2 :=root2+1/3*side*[cos(angle+Pi/6.),sin(angle+Pi/6.)]:
  root3:=root +1/6*side*[cos(angle),sin(angle)]:
  end3 :=root3+1/3*side*[cos(angle-Pi/6.),sin(angle-Pi/6.)]:
  graph:=[op(graph),line(root,end1),line(root2,end2),line(root3,end3)]:
  if n>=1 then
    _branch(end1,.7*side,angle,n-1):
    _branch(end2,.5*side,angle+Pi/6.,n-1):
    _branch(end3,.5*side,angle-Pi/6.,n-1):
  end if:
```

```
15 end proc:
16
17 > cntr:=[0,0]:
18 graph:=[]:
19 _branch(cntr,80.,Pi/2.,0):
20 display(graph,scaling=constrained,axes=none);
```

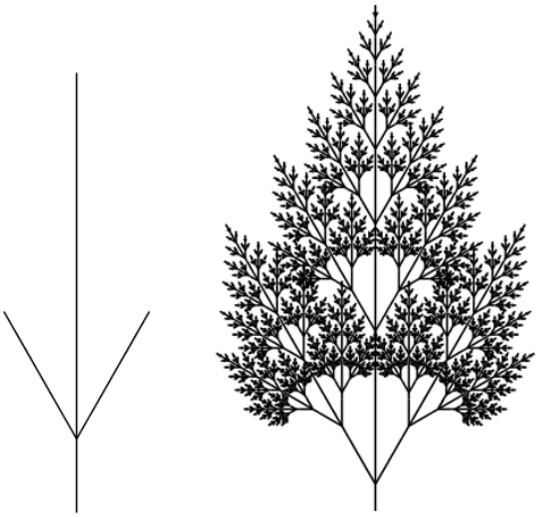


Рис. 11. Фрактальная ветка.

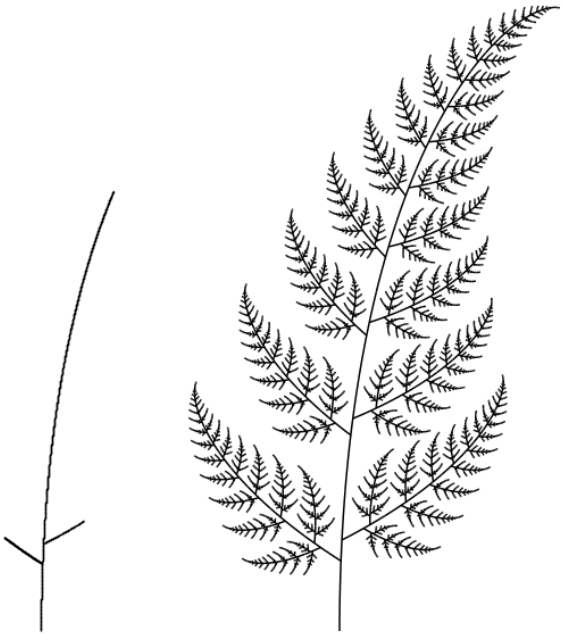


Рис. 12. Фрактальный папоротник.

Фрактальный папоротник

```
1  branch:=proc(root,l,angle,sym)
2  global graph:
3  local t1,t2,t3,r,r_,rotMatr,dr1,dr2,dr3:
4    t1:=.1:
5    t2:=.13:
6    t3:=.2:
7    r := [-.1411189523*cos(3.+.4*t)+0.10665e-5-.9899926471*sin(3.+.4*t),
         -.9899926471*cos(3.+.4*t)-1.000000000+.1411189523*sin(3.+.4*t)];
8   rotMatr:=array(1..2,1..2,[[cos(angle),-sin(angle)],[sin(angle),cos(angle)]]):
9    r_:=root+l*sym*convert(evalm(rotMatr&*subs(t=sym*t,r)),list):
10   graph:=[op(graph),plot([op(r_),t=0..t3],color=black)];
11   dr1:=evalf(subs(t=t1,diff(r_,t)),5):
12   dr2:=evalf(subs(t=t2,diff(r_,t)),5):
13   dr3:=evalf(subs(t=t3, diff(r_,t)),5):
14   evalf([
15     subs(t=t1,r_),
16     sign(dr1[2])*arccos(dr1[1]/sqrt(dr1[1]^2+dr1[2]^2)),
17     subs(t=t2,r_),
18     sign(dr2[2])*arccos(dr2[1]/sqrt(dr2[1]^2+dr2[2]^2)),
19     subs(t=t3,r_),
20     sign(dr3[2])*arccos(dr3[1]/sqrt(dr3[1]^2+dr3[2]^2))
21   ]);
22 end proc:
```

```
23
24 _fern:=proc(root,side,angle,sym)
25 global graph:
26 local temp:
27   if side > 1 then
28     temp:=branch(root,side,angle,sym):
29     _fern(temp[1],.35*side,temp[2]+sym*Pi/3.,sym):
30     _fern(temp[3],.35*side,temp[4]-sym*Pi/3,-sym):
31     _fern(temp[5],.8*side,temp[6],sym):
32   end if:
33 end proc:
34
35 > center:=[0,0]:
36 graph:=[]:
37 _fern(center,150.,Pi/2.,1);
38 display(graph,scaling=constrained,axes=none);
```

Литература

1. Кирсанов М.Н. Maple и Maplet. Решения задач механики. Учебное пособие/ М.Н. Кирсанов. – СПб: Лань, 2012. – 512 с.

GEOMETRIC FRACTALS CONSTRUCTION IN CAS MAPLE

A.A. Agathonov

Examples of constructing some known geometric fractals in the CAS Maple are given.

Keywords: geometric fractals, programming, CAS Maple.

УДК 378.147

ИСПОЛЬЗОВАНИЕ СИСТЕМ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ В РЕШЕНИИ ЗАДАЧ ТЕНЗОРНОГО АНАЛИЗА

А.В. Букушева[1]

1 *bukusheva@list.ru*; Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского

Рассматривается применение системы Wolfram Mathematica в решении задач тензорного анализа.

Ключевые слова: бакалавриат, Wolfram технологии, компьютерная геометрия, тензорный анализ.

Компьютерные технологии становятся регулярной, обязательной частью математического образования. Повышение качества процесса обучения математике обеспечивается за счет реализации дидактических возможностей информационных и коммуникационных технологий: автоматизации информационно-поисковой и вычислительной деятельности; моделирование, виртуальное представление на экране изучаемых математических объектов; расширения самостоятельной деятельности в условиях использования систем компьютерной математики (С.С. Кравцов, Л.П. Мартиросян, И.В. Роберт и др.).